UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

CPDLC – CONTROLLER PILOT DATA LINK COMMUNICATION

Rui Pedro Bento Ferreira

Versão Pública

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2011

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

CPDLC – CONTROLLER PILOT DATA LINK COMMUNICATION

Rui Pedro Bento Ferreira

PROJECTO

Trabalho orientado pelo Prof. Doutor António Manuel Silva Ferreira
e co-orientado por Eng. Manuel António Sousa Dias

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2011

Resumo

Com o aumento de tráfego aéreo que tem ocorrido nos últimos anos e com o que se prevê que venha a acontecer num futuro próximo é necessário começar a tomar medidas para que os sistemas de controlo de tráfego aéreo suportem este aumento. Estas medidas também têm como objectivo evitar que os custos aumentem com o aumento de tráfego aéreo bem como ultrapassar a escassez cada vez maior de frequências rádio.

É neste âmbito que surge uma nova tecnologia denominada de *Controller Pilot Data Link Communication* (CPDLC, esta tecnologia pretende substituir a comunicação por voz efectuada entre o piloto e o controlador de tráfego aéreo por comunicação por texto, tratando-se de um serviço de mensagens escritas para situações não críticas e aplicável a aeronaves acima de determinada altitude (altitude de rota).

Este relatório descreve o evoluir de um simulador de controlo de tráfego aéreo existente, designado SIMATM, para que este suporte a nova funcionalidade CPDLC. A realização do projecto seguiu o modelo de desenvolvimento de *software* em V usado na NAV Portugal, E.P.E., que se traduziu na adição de um simulador de *Data Link Server* ou SIMDLS, criado de raiz, sendo que a sua necessidade foi decidida no decurso do projecto, durante a análise do problema, bem como na evolução das componentes internas do SIMATM.

Resultante da realização do projecto foi também a produção e publicação de um artigo científico para a 6ª Conferência Ibérica de Sistemas e Tecnologias de Informação.

Palavras-chave: Controlo de tráfego aéreo, Simulador, Comunicação, *Data Link*, CPDLC

Abstract

Due to the increase of air traffic that has occurred in recent years and that is expected to continue in the near future, it is necessary to take measures to ensure air traffic control systems can keep up with this tendency. These measures also aim to prevent cost escalation as well as deal with the increasing scarcity of radio frequencies.

It is in this context that a new technology called Controller Pilot Data Link Communication (CPDLC) emerges, which aims to replace voice communications between pilots and air traffic controllers with a text messaging service for non-critical situations, applicable to aircrafts above a certain altitude.

My project aimed at augmenting an existing simulator for air traffic control, called SIMATM, to support this new CPDLC functionality. The execution of this project followed the V-model of software development that has been adopted by NAV Portugal, E.P.E. The main deliverables are, on the one hand, a new Data Link Server simulator, or SIMDLS, whose development was decided during the problem analysis phase, and, on the other hand, the evolution of the internal components of the SIMATM simulator so that it supports CPDLC.

Another deliverable of this project is the publication of a scientific paper in the 6th Iberian Conference on Information Systems and Technologies.

Keywords: Air Traffic Management, Simulator, Communication, Data Link, CPDLC

Índice

Lista de figuras

Lista de tabelas

Lista de acrónimos

Acrónimo	Significado
ACM	*Air Communication Management*
ADI	*ATM/Data Link Interface*
ANSP	*Air National Service Provider*
ASD	*Air Situation Display*
ATC	*Air Traffic Control*
ATCC	*Air Traffic Control Center*
ATCO	*Air Traffic Control Operator*
ATFC	*Aircraft Track Flight Coupling*
ATM	*Air Traffic Management*
ATN	*Air Traffic Network*
ATLATM	*Atlantic Air Traffic Management System*
ASN.1	*Abstract Syntax Notation 1*
CPDLC	*Controller Pilot Data Link Communication*
DLP	*Data Link Processor*
DLS	*Data Link Server*
DSTI	Direcção de Sistemas e Tecnologias de Informação
EUROCONTROL	*European Organization for the Safety of Air Navigation*
FIR	*Flight Information Region*
FPL	*Flight Plan*
GS	*Game Supervisor*
GSD	*Ground Situation Display*
HMI	*Human Machine Interface*
ICAO	*International Civil Aviation Organization*
IFPS	*Initial Flight Plan Processing System*
LISATM	*Lisbon Air Traffic Management System*
NAV	Navegação Aérea de Portugal
PP	*Pseudo Pilot*
SIMATM	*Simulator for Air Traffic Management System*
SIMDLS	Simulador de *Data Link Server*

SISINT	Sistemas – Interface com o Utilizador
SISLOG	Sistemas – Logística
SISPRO	Sistemas – Produção
SISQUA	Sistemas – Qualidade
TCP	*Transmission Control Protocol*
TWRATM	*Tower Air Traffic Management System*
UDP	*User Datagram Protocol*
XML	*Extensible Markup Language*
xmlRAC	*Xml Register-Alive Connection*

Capítulo 1

Introdução

Este documento tem o propósito de descrever o trabalho que fiz na empresa NAV Portugal, E.P.E. durante a realização do Projecto de Engenharia Informática. Este trabalho teve como objectivo geral implementar um novo serviço de mensagens escritas num simulador de tráfego aéreo existente.

1.1 Motivação

O volume de tráfego aéreo tem vindo a aumentar em todo o mundo ao longo dos últimos anos [1] e esta tendência deverá continuar no futuro a uma taxa anual de 3%, o que significa que em 2030 haverá quase o dobro do número de movimentos de voo do que hoje [5]. Esta situação irá provocar problemas na gestão do tráfego aéreo, isto porque as frequências rádio utilizadas hoje em dia para fazer a comunicação piloto – controlador são um recurso cada vez mais escasso. Uma razão para tal é que quando o número de aeronaves que voam num determinado sector de controlo (área geográfica) atinge um limite, o sector é dividido em dois, necessitando de uma frequência rádio extra atribuída a um novo controlador.

Naturalmente, um problema do congestionamento de frequências rádio é que mais pessoas necessitam de compartilhar o mesmo canal de voz, aumentando assim o risco de problemas de comunicação, que são uma fonte potencial de erros operacionais, devido ao ruído e interferências, comunicação ambígua, complexidade das mensagens e outros factores [15]. Além disso estas confusões existentes entre pilotos e controladores são normalmente resolvidas por repetição da mensagem, o que leva a uma tendência de congestionamento de frequências ainda maior.

Para fazer face à escassez de frequências rádio e aumentar a capacidade operacional dos prestadores de serviços de tráfego aéreo nacionais, ou ANSP (*Air National Ser-*

vice Provider)—responsáveis pelo fluxo seguro, rápido e ordenado do tráfego aéreo—a *International Civil Aviation Organization* (ICAO) e a Comissão Europeia, através da *European Organisation for the Safety of Air Navigation* (EUROCONTROL) aprovaram a tecnologia *Controller Pilot Data Link Communication* (CPDLC), que oferece um serviço de mensagens escritas fazendo uso de comunicação digital para conectar as aeronaves e os centros de controlo através da rede de telecomunicações aeronáuticas, ou ATN (*Air Traffic Network*) [18].

A tecnologia CPDLC deverá substituir gradualmente algumas das mensagens de voz de rotina, que não sejam críticas trocadas entre piloto e controlador durante as operações no ar como por exemplo a alteração do nível de voo. A seguir salientam-se algumas das principais vantagens da utilização de CPDLC no controlo de tráfego aéreo, conforme reportado na literatura:

- Aumento da capacidade do canal de comunicação devido a serem necessários menos *bits* para transferir as mensagens de texto do que as mensagens de voz equivalentes [17]. Além disso, pedidos de retransmissão provenientes dos pilotos ou dos controladores são praticamente desnecessários, porque uma parte significativa dos erros em mensagens digitais podem ser automaticamente detectados e corrigidos [14];
- Aumento da capacidade e produtividade de cada sector de controlo, devido à possibilidade das aeronaves enviarem periodicamente mensagens de relatórios de estado que de outra maneira teriam de ser transmitidas por voz [8], e também devido à simplificação dos procedimentos de rotina, por exemplo, atribuir mensagens recorrentes a botões dedicados numa interface pessoa-máquina [23]. A redução da necessidade de comunicação verbal tem sido associada à menor carga de trabalho do controlador, o que significa que mais cerca de 15% das aeronaves podem ser controladas em segurança em cada sector [17].

Ambas as vantagens apresentadas mitigam o problema da escassez de frequências rádio através da melhor utilização dos recursos existentes. Para enfatizar ainda mais a importância da tecnologia CPDLC no controlo de tráfego aéreo, a Comissão Europeia, através da EUROCONTROL, aprovou uma regra obrigatória que diz que todas as aeronaves construídas desde 2011 têm de estar equipadas com o serviço de comunicação digital, aplicando-se a mesma regra a partir de 2013 a todos os ANSPs europeus.

Este documento descreve todo o trabalho realizado para implementar o serviço CPDLC no simulador de controlo de tráfego aéreo existente na NAV Portugal E.P.E., simulador esse que é utilizado para treino de controladores e teste de novas funcionalidades introduzidas no sistema de controlo de tráfego aéreo.

1.2 Objectivos

Este trabalho teve três objectivos principais descritos de seguida:

1. Implementar um protótipo para a utilização de mensagens CPDLC no Simulador de Sistema de Controlo de Tráfego Aéreo – SIMATM. Para tal é necessário concretizar primeiro a ligação *Data Link* entre o SIMATM e o Sistema de Controlo de Tráfego Aéreo – LISATM;
2. Evoluir o ASD (*Air Situation Display*) para o sistema LISATM (*Lisbon Air Traffic Management System*) para que este possa receber as mensagens CPDLC e apresentá-las na interface com o utilizador permitindo também ao utilizador poder responder às mensagens CPDLC provenientes do SIMATM, bem como enviar novas mensagens;
3. Realizar um protótipo para o GSD (*Ground Situation Display*) de forma a se substituir a comunicação por voz por comunicação digital também no solo, como por exemplo autorizações de partida. O GSD fornece ao utilizador uma vista radar do que se passa no solo.

O desenvolvimento destes protótipos teve como base uma análise de requisitos bem como uma análise da arquitectura já existente de forma a se perceber como alterar os sistemas em causa.

A implementação é baseada nestas análises e no desenho efectuado, sendo também necessário planear e executar testes para todos estes protótipos e para todas as funcionalidades implementadas.

1.3 Instituição de acolhimento

A empresa onde desenvolvi o meu Projecto em Engenharia Informática é a NAV Portugal, E.P.E, que tem como missão prestar os serviços de controlo de tráfego aéreo nas regiões de informação de voo (FIR – *Flight Information Region*) de Lisboa e de Santa Maria, garantindo o cumprimento da regulamentação nacional e internacional.

A NAV Portugal E.P.E. tem definido os seguintes pontos na sua missão:

- Optimização dos padrões de segurança do tráfego aéreo;
- Resposta adequada à procura do tráfego;
- Melhoria da relação eficiência/custo;
- Evolução para a excelência, no âmbito da gestão da qualidade.

Depois de apresentada a empresa de acolhimento passo a apresentar a DSTI (Direcção de Sistemas e Tecnologias de Informação) na qual o meu estágio decorreu.

A DSTI é responsável pelo desenvolvimento de *software* para suporte ao controlo de tráfego aéreo e é constituída por 4 departamentos:

- SISQUA: Responsável pela área de qualidade e *safety*;
- SISLOG: Responsável pela logística;
- SISINT: Responsável pela interface com o utilizador;
- SISPRO: Responsável pela produção de S*oftware*.

Este estágio desenvolveu-se em primazia no serviço SISPRO sob supervisão do Eng. José dos Santos Mestre Vermelhudo.

1.4 Planeamento e execução

Os objectivos apresentados na Secção 1.2 são os mencionados (os mais relevantes encontram-se a negrito) na lista inicial de tarefas do projecto datada de Outubro de 2010, como a seguir apresentado:

ID	Descrição	Duração (dias)
E1	Formação no sistema LISATM	1
E2	Introdução ao subsistema ASD da NAV	4
E3	Introdução ao Sistema de Gestão de Qualidade	1
E4	Preparar ambiente de trabalho e ferramentas de desenvolvimento	4
E5	**Desenvolvimento de protótipo inicial de implementar comandos CPDLC no SIMATM**	**49**
E6	**Desenvolvimento de protótipo *Data Link* no ASD**	**39**
E7	**Desenvolvimento de protótipo para o GSD**	**30**
E8	Produção de relatório preliminar	5

E9	Produção de relatório final em versão pública	5
E10	Produção de relatório final em versão confidencial	30

Tabela 1: Lista inicial de tarefas do projecto

Com a decisão apresentada no Capítulo 3, foi necessário efectuar uma reestruturação das tarefas de forma a conseguir superar as novas exigências encontradas.

Sendo assim, no mês de Fevereiro de 2011 foi tomada a decisão de como seria a interface de dados *data link* entre o simulador e o sistema de controlo de tráfego aéreo, levando a que fosse concretizada uma nova componente para o simulador, componente essa com o nome de Simulador de Servidor de *Data Link* (SIMDLS). Sendo assim das tarefas presentes na Tabela 1 foram eliminadas a tarefa E6 e E7 sendo substituídas por uma tarefa denominada de Desenvolvimento do SIMDLS, identificada como E6 na lista revista de tarefas apresentada na Tabela 2.

ID	**Descrição**	**Duração (dias)**
E1	Formação no sistema LISATM	1
E2	Introdução ao subsistema ASD da NAV	4
E3	Introdução ao Sistema de Gestão de Qualidade	1
E4	Preparar ambiente de trabalho e ferramentas de desenvolvimento	4
E5	**Desenvolvimento de protótipo inicial de implementar comandos CPDLC no SIMATM**	**83**
E6	**Desenvolvimento do SIMDLS**	**35**
E7	Produção de relatório preliminar	5
E8	Produção de relatório final em versão pública	5
E9	Produção de relatório final em versão confidencial	30

Tabela 2: Lista final de tarefas do projecto

Uma explicação mais aprofundada das razões que levaram a esta alteração na lista de tarefas do projecto pode ser encontrada no Capítulo 3.

Tendo em conta o planeamento apresentado na Tabela 2, e finalizado o projecto é importante dar ênfase ao que foi efectivamente executado. Em termos da duração atribuída às tarefas houve uma tentativa de cumprir os dias estipulados, sendo que as tare-

fas E5 e E6 acabaram por ter uma duração menor por não ter sido completado o ciclo de desenvolvimento de software, como é referido no Capítulo 4. A produção de um artigo científico também retirou algum tempo às tarefas de desenvolvimento, pois foi necessário traduzir para inglês algum material vindo do relatório preliminar bem como produzir conteúdos novos. As restantes tarefas presentes na Tabela 2 foram cumpridas com sucesso não tendo existido desvios assinaláveis nos dias atribuídos.

1.5 Principais resultados

Os principais resultados obtidos da realização deste projecto foram os seguintes:

1. Evolução do SIMATM para suportar a capacidade de CPDLC. Esta evolução permite que o simulador de controlo de tráfego aéreo possa enviar e receber mensagens CPDLC, bem como adiciona uma nova componente funcional ao SIMATM denominada de Simulador de Servidor de *Data Link*. Em termos das componentes que já existiam no SIMATM o trabalho passou por evolui-las para suportarem esta nova funcionalidade. Já o SIMDLS é uma nova componente adicionada ao simulador e que tem o objectivo de simular uma componente que existe no sistema LISATM;
2. Produção de um artigo científico, em anexo, com o título de "*Developing a Controller Pilot Data Link Communication Simulator*" apresentado na 6ª Conferência Ibérica de Sistemas e Tecnologias de Informação [6] que decorreu em Chaves em Junho de 2011, o qual será divulgado na biblioteca digital do IEEE.

1.6 Notação adoptada

De forma a contextualizar o leitor é importante referir que ao longo deste documento são utilizados termos tanto em inglês como em português. Os termos em inglês, como por exemplo *Pseudo Pilot* e *Scenario Editor*, são utilizados para referir componentes funcionais. Os termos em português, como por exemplo Pseudo Piloto e Criador de cenário, referem-se aos operadores das componentes funcionais em causa. Adicionalmente, outros termos em inglês também são colocados em itálico ao longo do documento.

1.7 Organização do documento

O restante deste relatório está organizado da seguinte maneira:

- Capítulo 2 – Trabalho relacionado: Este capítulo aborda o material existente na literatura académica relacionado com o CPDLC, bem como descreve os sistemas existentes na NAV que estão directamente ligados com o meu projecto, nomeadamente o sistema utilizado no controlo de tráfego aéreo, LISATM, e o sistema que permite simular cenários de tráfego aéreo, SIMATM;
- Capítulo 3 – Abordagem ao problema: Este capítulo pretende apresentar a abordagem efectuada ao problema, sendo dada ênfase às alternativas existentes para a interface entre os sistemas SIMATM e LISATM e à decisão que foi tomada. Também apresento neste capítulo os conceitos que são mais importantes para uma melhor percepção do trabalho efectuado;
- Capítulo 4 – Trabalho realizado: Neste capítulo abordo todo o trabalho de desenvolvimento de *software* realizado no âmbito do projecto. Esse trabalho é dividido em dois grandes grupos: a concretização do Simulador de Servidor de *Data Link* e a evolução do SIMATM para poder processar mensagens CPDLC.
- Capítulo 5 – Conclusão: Por fim apresento as conclusões do trabalho, sendo referidas as principais contribuições, as dificuldades encontradas, competências adquiridas, e uma perspectiva de trabalho futuro.

Capítulo 2

Trabalho relacionado

É abordado neste capítulo algum trabalho já efectuado acerca do CPDLC, mais precisamente alguns estudos que podem ser encontrados na literatura académica.

Este capítulo pretende também introduzir o leitor aos sistemas já existentes na NAV Portugal E.P.E. e que têm ligação com o meu trabalho, dando a conhecer onde este se enquadra.

2.1 CPDLC na literatura académica

A tecnologia CPDLC (*Controller Pilot Data Link Communication*) tem vindo a amadurecer ao longo dos últimos anos e capturando o interesse de investigadores que utilizam simulação computacional para diversas tarefas, tais como:

- Avaliação de carga cognitiva baseada na execução de tarefas em cenários de comunicação pré-definidos, que previram desequilíbrios na carga de trabalho entre piloto e co-piloto [11] e melhorias da eficiência do controlador em ambiente misto visual/auditivo [23];
- Avaliação de usabilidade através de protótipos de alta e baixa fidelidade que reproduzem o fluxo de tráfego aéreo para identificar problemas na interacção pessoa-máquina [10], como o ecrã ficar uma confusão quando o CPDLC é integrado numa ferramenta de apoio à decisão [20] ou o aumento do risco dos controladores executarem acções redundantes quando as mensagens de texto substituem a comunicação por voz [9];
- Avaliação de performance de rede através da introdução de mensagens CPDLC no protocolo de comunicação e verificação que os requisitos de tempo de transferência continuam válidos [19], para demonstrar que um protocolo é mais efi-

ciente que outro [16], ou para mostrar que a capacidade da ligação de dados proposta consegue suportar o tráfego futuro [21];

- Avaliação operacional feita aceitando comandos tanto do piloto como do controlador durante a simulação, sendo encontrados na literatura três simuladores:
  - *Advanced Communications for* ATM – AC/ATM [12]: Para estimar o número máximo de aviões que podem operar em segurança numa única frequência rádio, com maior ênfase para o simulador suportar 160 aeronaves em simultâneo [12].
  - *User Requested Evaluation Tool with* CPDLC – URET/CPDLC [3]: Investigação sobre alterações de usabilidade reportadas aquando da integração de CPDLC na ferramenta de suporte à decisão URET existente [3].
  - *Communication, Navigation, Surveillance for* ATM – CNS/ATM [8]: Aceder ao CPDLC numa rede de tráfego aéreo em condições extremas, para auferir o grau de confiança no mesmo.

Estes três simuladores não são concorrentes directos do simulador SIMATM existente na NAV Portugal E.P.E., pois, ao contrário do SIMATM que tem como foco aproveitar a sinergia treino/teste, a mais-valia destes três simuladores é a de testar as redes de comunicação em situações extremas (quantas aeronaves podem ser acompanhadas ao mesmo tempo) e avaliar a integração do CPDLC numa ferramenta já existente de apoio à decisão.

2.2 Trabalho relacionado da NAV

Esta secção divide-se em duas e pretende descrever ao leitor dois sistemas existentes na NAV Portugal E.P.E. que estão ligados com o meu trabalho. Sendo assim é descrito o sistema de controlo de tráfego aéreo, LISATM, e o simulador para sistema de controlo de tráfego aéreo, SIMATM.

2.2.1 *Lisbon Air Traffic Management System*

O LISATM (*Lisbon Air Traffic Management System*) é o sistema de controlo de tráfego aéreo existente no centro de controlo de Lisboa. Versões deste sistema também se encontram presentes nas diversas torres de controlo do continente e do arquipélago da Madeira, bem como em Santa Maria nos Açores, sendo estas duas últimas denominadas

TWRATM (*Tower Air Traffic Management System*) e ATLATM (*Atlantic Air Trafic Management System*).

Sendo o LISATM o sistema de suporte ao controlo de tráfego aéreo da FIR (*Flight Information Region*) Portuguesa, é da sua responsabilidade disponibilizar uma interface ao controlador com uma vista radar do espaço aéreo nacional. Além desta vista radar o controlador tem também disponível informação sobre as aeronaves como por exemplo identificação, posição, velocidade, altitude, plano de voo, rota, entre outros dados como informação meteorológica e dos aeroportos.

Pela descrição apresentada percebe-se que todo o funcionamento do sistema LISATM é bastante complexo, sendo que para obter todas as informações necessárias é preciso interagir com diversas entidades externas ao sistema. Essas entidades são responsáveis por fornecer ou receber informação do mais variado tipo como por exemplo planos de voo, informação de vigilância (radar, etc), e número de voos realizados e sua duração, entre outros.

A Figura 1 mostra um diagrama de contexto do LISATM onde se pode verificar a grande quantidade de entidades que interagem com este sistema, sendo dada ênfase à entidade SIMATM, pois foi no contexto desta que realizei o meu trabalho.

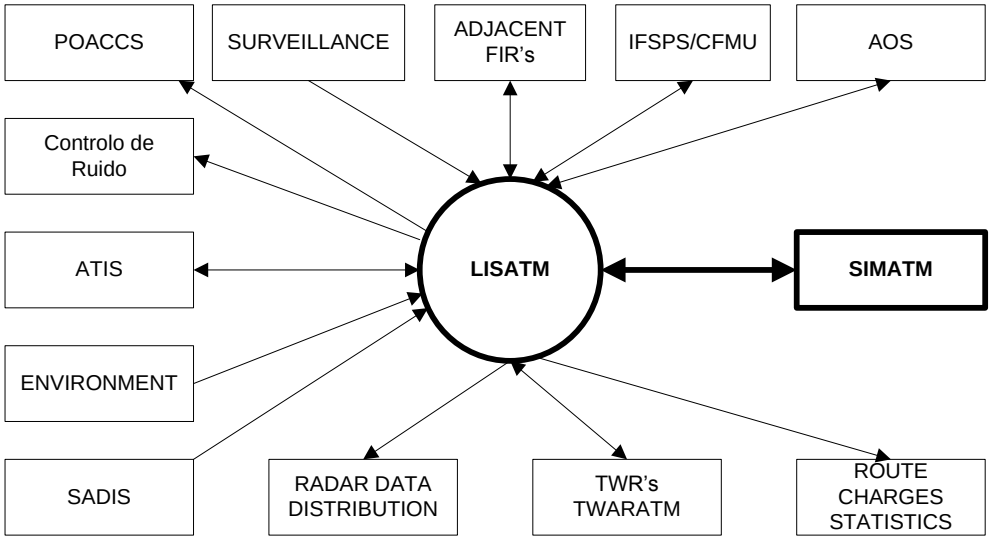

Figura 1: Diagrama de contexto do LISATM

Sendo o LISATM tão complexo é natural que seja constituído por um grande número de componentes, cada uma delas com a sua função específica dentro do sistema e dando o seu contributo para que este possa funcionar sem problemas. Também é importante salientar que todas as componentes do LISATM são consideradas críticas,

pois a falha de uma pode gerar uma situação de degradação que pode ser grave tendo em conta o contexto em que o sistema é usado. Assim, de forma a evitar essas situações em caso de falha, todas as componentes críticas do LISATM estão replicadas garantindo a necessária redundância para que seja sempre possível desempenhar a sua função.

A Figura 2 mostra que o SIMATM tem a função de simular todos os dados que o LISATM poderia receber dos sistemas externos, incluído as aeronaves. Nestas circunstâncias o comportamento do LISATM quando está a receber dados de simulação mantém-se o mesmo.

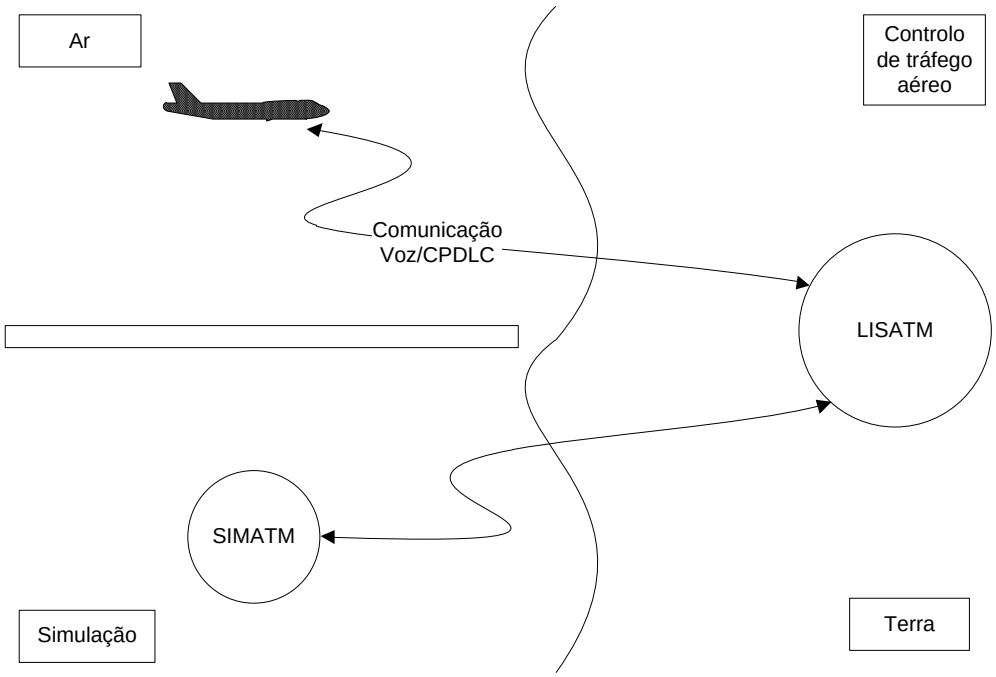

Figura 2: Ligação entre sistemas ar, terra, e simulação.

O meu trabalho pretende acrescentar a funcionalidade CPDLC no SIMATM para que se consiga, através de simulação, enviar este novo tipo de mensagens para o LISATM.

2.2.2 *Simulator for Air Traffic Management System*

O SIMATM é um simulador desenvolvido pela NAV Portugal E.P.E. para aproveitar a sinergia entre treino de controladores e testes de novas funcionalidades [7]:

- Por um lado, os controladores em treino têm acesso a uma réplica do sistema de controlo de tráfego aéreo LISATM. Deste modo o treino é realizado nos mesmos moldes do que acontece na realidade;

- Por outro lado, os eventos gerados pelos controladores em treino enquanto interagem com o LISATM podem fazer parte dos testes do sistema, complementando a utilização de cenários estáticos e pré-definidos.

Nos simuladores para sistemas de controlo de tráfego abordados na Secção 2.1 esta sinergia não é totalmente aproveitada (em alguns casos nem existe) porque os controladores humanos operam protótipos de alta/baixa fidelidade em vez de sistemas reais.

O desenvolvimento SIMATM começou no fim do ano de 2004 sendo que a primeira instalação operacional entrou em funcionamento em Agosto de 2006, sendo que a versão actual é a 2.0.

A Figura 3 é baseada num diagrama de fluxo de dados do documento de especificação da arquitectura do SIMATM [13], sendo aqui apresentados apenas os fluxos mais relevantes para explicar o seu funcionamento e para melhor enquadrar o meu trabalho.

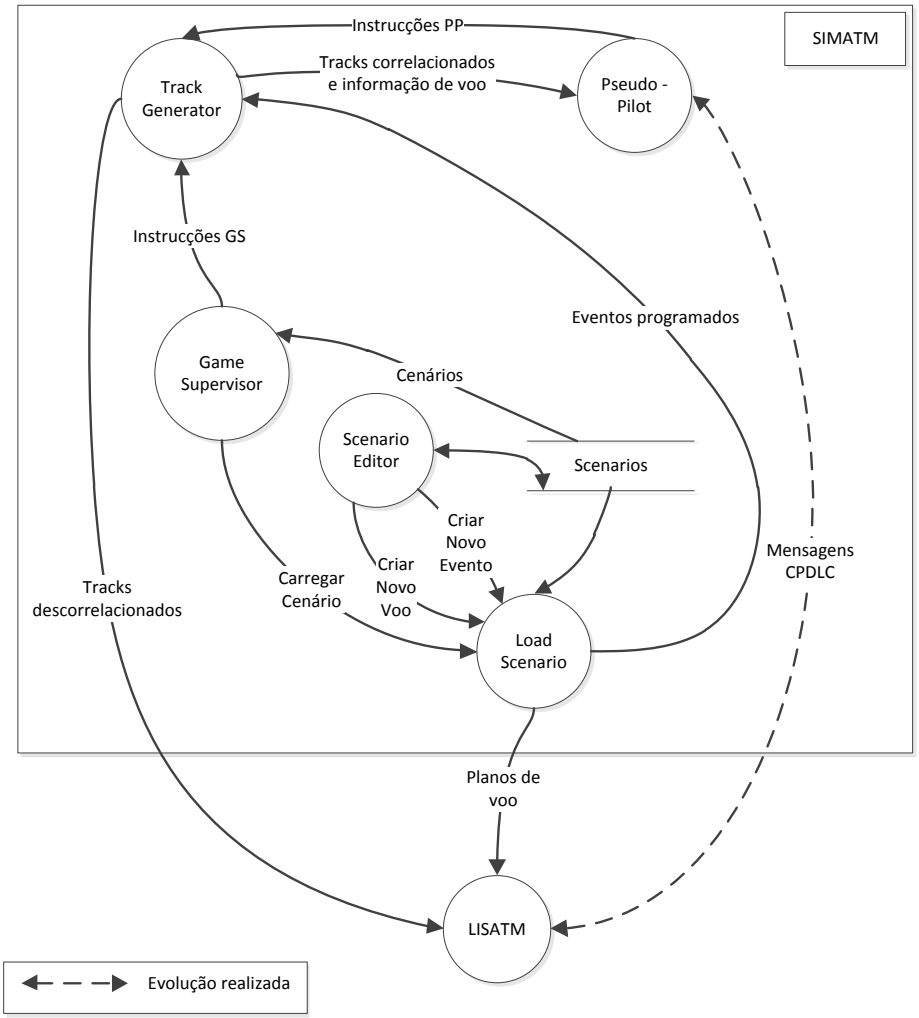

Figura 3: Diagrama de fluxo de dados entre SIMATM e o LISATM

Neste diagrama podem ver-se cinco das componentes funcionais do SIMATM sendo que de seguida faço uma breve descrição de cada uma:

- *Scenario Editor*: Possibilita criar ou actualizar um cenário de exercício simulado. O cenário criado é guardado num ficheiro em formato XML sendo posteriormente armazenado numa base de dados de cenários;
- *Load Scenario*: Responsável por converter um voo presente no ficheiro do cenário num plano de voo (*Flight Plan*) a ser enviado ao sistema LISATM e utilizado pelo *Track Generator* e *Pseudo Pilot*;
- *Track Generator*: Calcula a próxima posição das aeronaves com base na informação contida no cenário, nos comandos do *Pseudo Pilot* e nos comandos do *Game Supervisor*. O resultado é enviado como informação de vigilância. Esta entidade é responsável, entre outras coisas, por alimentar o LISATM com dados como se estes fossem provenientes dos radares;
- *Game Supervisor*: Permite o controlo manual do exercício (*load*, *play*, *pause*, *stop*), distribuição de aeronaves pelos *pseudo pilot* existentes, podendo também assumir o papel *pseudo pilot*;
- *Pseudo Pilot*: Permite controlar individualmente cada uma das aeronaves existentes no exercício. Esta componente permite também instruir a aeronave para ter um comportamento de piloto automático, no qual a navegação da aeronave se baseia na informação guardada no plano de voo.

Como se pode ver na Figura 3, algumas componentes do simulador fornecem dados ao sistema LISATM, tais como informação de vigilância proveniente do *Track Generator* e planos de voo do *Load Scenario*, os quais são importantes para que o comportamento do LISATM seja igual ao que realmente acontece.

Tendo em conta os três fluxos que entram no LISATM apresentados na Figura 3, o único que importa aprofundar a discução tendo em vista o âmbito do meu trabalho é o das mensagens CPDLC, mostrado no lado direito.

Supondo uma situação real, em que as mensagens CPDLC são provenientes da aeronave e não do simulador, então o caminho das mensagens desde o avião até ao HMI (*Human-Machine Interface*) do LISATM o apresentado na Figura 4:

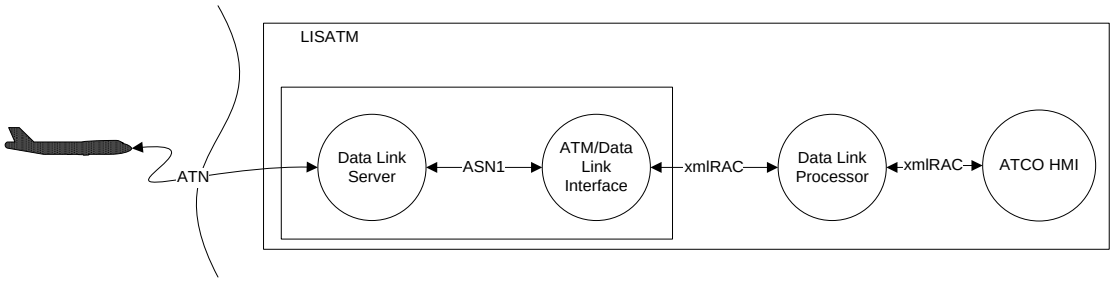

Figura 4: Entrada e saída de mensagens CPDLC no LISATM

Pode-se constatar também a constante mudança de protocolos de comunicação necessários para transmitir uma mensagem CPDLC através das várias componentes do LISATM.

Existe um protocolo de comunicação presente no diagrama que é importante referir, nomeadamente o xmlRAC. Este protocolo é baseado em XML e serve de base para várias ligações internas ao sistema. É implementado utilizando *sockets* UDP, mas que garante a entrega das mensagens a nível protocolar. O xmlRAC também é utilizado no SIMATM para a comunicação entre os seus nós.

2.3 Sumário

Neste capítulo abordei o trabalho relacionado com o CPDLC, tendo feito um resumo da literatura académica existente, que mostra que o CPDLC é uma tecnologia que tem vindo a amadurecer ao longo dos anos.

Também descrevi dois sistemas existentes na NAV Portugal E.P.E. que estão directamente ligados com o projecto em que me insiro: o sistema de controlo de tráfego aéreo, LISATM, e o sistema de simulação de controlo de tráfego aéreo, SIMATM.

O próximo capítulo tem como objectivo apresentar a abordagem ao problema proposto para o meu projecto, sendo dada ênfase à decisão de qual a interface a usar para trocar mensagens CPDLC entre o LISATM e o SIMATM.

Capítulo 3

Abordagem ao problema

Este capítulo pretende apresentar a abordagem inicial ao problema proposto de evoluir o SIMATM (*Simulator for Air Traffic Management System*) para que este suporte a funcionalidade CPDLC (*Controller Pilot Data Link communication*). Assim, começarei por enquadrar a tecnologia CPDLC no contexto do controlo de tráfego aéreo incluindo uma descrição dos principais conceitos, para de seguida apresentar as alternativas e decisões de arquitectura tomadas.

3.1 Enquadramento

Apresento agora uma breve descrição de como decorre a interacção piloto – centro de controlo, desde que o avião está no chão do aeroporto de partida até aterrar no aeroporto de chegada. Esta descrição é importante porque elucida o leitor de como funciona a comunicação sem recurso ao CPDLC.

Na Figura 5 pode ver-se um exemplo ilustrativo de um voo com algumas situações que permitem descrever como é feita a comunicação piloto – centro de controlo com a tecnologia existente. Neste exemplo um avião está para descolar do aeroporto A localizado numa determinada FIR (*Flight Information Region*) A, para o aeroporto B localizado na FIR B, sendo que durante o voo é feito um pedido de mudança de nível de voo.

No ponto 1 a aeronave encontra-se parada no aeroporto A e pretende obter autorização para descolar. Sendo assim, entra em contacto com a torre de controlo via rádio para requerer essa autorização. A torre, depois de verificar se é ou não possível aquela aeronave descolar, responde ao piloto via rádio e o piloto, caso a resposta seja positiva, dirige-se para a pista a fim de descolar.

De seguida, no ponto 2, o piloto começa por descolar mantendo contacto rádio com a torre de controlo do aeroporto A. Durante a subida o contacto é mantido com o controlador de aproximação deste aeroporto e quando está em nível de rota (finalizou a

descolagem), o contacto é feito com o controlador do sector de rota, adjacente ao sector de aproximação. Estes dois últimos passos são já realizados com controladores presentes no ATCC (*Air Traffic Control Center*). Para simplificar, estes dois últimos controladores não são mostrados na Figura 5.

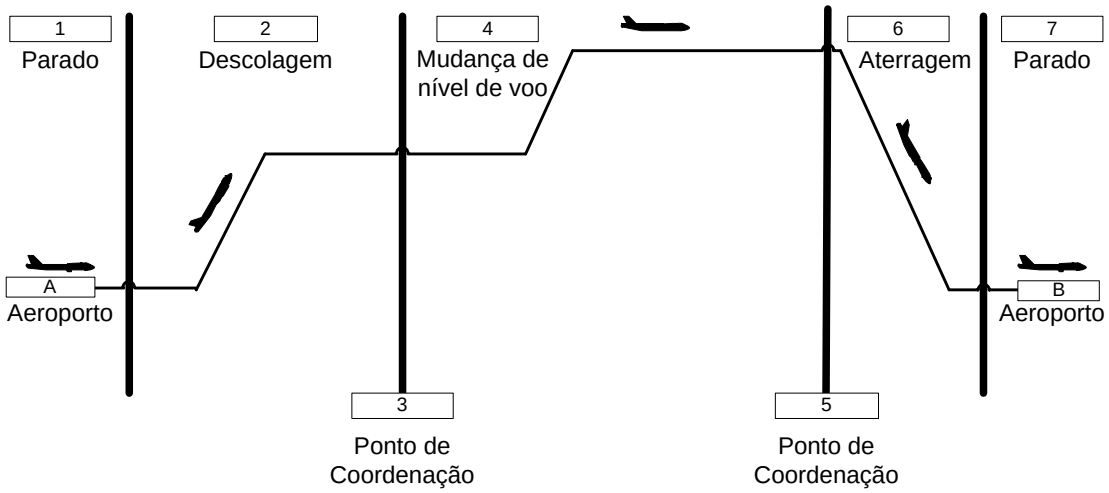

Figura 5: Cenário de fases de um voo

O ponto 3 indica um ponto de coordenação, no qual o controlador responsável pela aeronave muda e onde tem de haver uma coordenação para que o piloto fique a falar com a pessoa/frequência certa. Essa coordenação pode ser entre controladores de sectores de controlo diferentes dentro da mesma FIR, como acontece no ponto 2, ou de FIRs diferentes como neste caso, sendo efectuada da seguinte forma:

- O controlador responsável na FIR A entra em contacto com o controlador que vai ficar responsável pela aeronave na FIR B informando que daqui a *m* minutos irá receber o avião TAP123, num determinado nível de voo, num determinado ponto de coordenação sendo que esta é uma das formas de identificar a aeronave;
- O controlador da FIR B valida ou invalida a recepção do avião TAP123;
- Caso a recepção seja válida, o controlador da FIR A entra em contacto com o piloto do TAP123 indicando que a partir do ponto de coordenação irá estar sobre a alçada da FIR B e que deve contactar a FIR B na frequência *f*;
- O piloto responde ao controlador da FIR A e entra em contacto com o controlador da FIR B, estando feita a coordenação entre a FIR A e a FIR B.

Esta é uma fase delicada que exige muito uso da frequência de rádio havendo o risco, de congestionamento de frequência caso seja necessário vária repetições.

Já sob a alçada do controlador da FIR B, o piloto necessita de pedir uma alteração do nível de voo, visível no ponto 4, pelo que entra mais uma vez em contacto com o controlador informando-o do seu pedido e o controlador responde consoante a situação. Caso a resposta seja positiva então é alterado o nível de voo.

Por fim, a aproximação à pista, no ponto 5, corresponde a mais um ponto de coordenação, mas este dentro da mesma FIR. Esta coordenação é feita entre o controlador responsável pelo avião e o controlador responsável pela aproximação à pista. Por fim, para aterrar o avião (ponto 6) o piloto entra em contacto com o controlador para obter a autorização de aterragem, e caso esta seja positiva é feita a aterragem e a nível de controlo de tráfego aéreo o voo está terminado.

As várias fases de um voo mostradas na Figura 5 podem ser simuladas integralmente com o SIMATM constituindo aquilo que se designa por cenário de exercício. Um exercício corresponde à execução de um cenário, sendo que nestas circunstâncias se está a exercitar o LISATM com dados de simulação em vez de dados reais.

Como se pode ver por esta descrição resumida o processo de comunicação entre piloto e controlador que decorre durante um voo é complexo sendo este o principal motivo para a adopção da função CPDLC. Tendo este trabalho como objectivo principal implementar mensagens CPDLC no SIMATM é importante primeiro que tudo uma explicação do que consiste o CPDLC.

Hoje em dia toda a comunicação trocada entre piloto e controlador é efectuada recorrendo a frequências rádio, o que, com o aumentar do tráfego aéreo leva a uma cada vez maior congestão dessa frequência e consequente desdobramento, levando assim à escassez de frequências disponíveis no espectro utilizado na aviação civil, tal como mencionado na Secção 1.1. É devido a este problema que surge a tecnologia *Data Link*, que tem como objectivo criar um canal de comunicação digital entre o piloto e o controlador, mitigando desta forma a escassez de frequências rádio e reduzindo o custo em recursos. A *Controller Pilot Data Link Communication* funciona recorrendo a este canal de comunicação digital, e tem como objectivo substituir as mensagens de voz para alteração de perfil de voo que não sejam críticas trocadas entre piloto e controlador por mensagens semelhantes mas em formato de texto; no fundo acaba por ser um serviço de mensagens escritas entre piloto e controlador e vice-versa.

As mensagens CPDLC encontram-se previamente definidas, formatadas, e normalizadas sendo que o conjunto de mensagens no contexto do meu trabalho é o que consta na Tabela 3.

Direcção	**Nome**	**Descrição**
Controlador para Piloto	MANTAIN [*level*]	Manter nível de voo [*level*]
	CLIMB TO [*level*]	Subir para nível de voo [*level*]
	DESCEND TO [*level*]	Descer para nível de voo [*level*]
Piloto para Controlador	REQUEST MANTAIN [*level*]	Semelhante às mensagens acima mas iniciadas pelo piloto. O piloto necessita esperar a resposta do controlador
	REQUEST CLIMB TO [*level*]	
	REQUEST DESCEND TO [*level*]	
	WILCO	Mensagem compreendida e será executada
Ambas as direcções	UNABLE	Não posso executar
	STANDBY	Resposta será enviada

Tabela 3: Lista de mensagens CPDLC implementadas neste trabalho

3.2 Alternativas de arquitectura

Depois de descritos os conceitos principais, passo a abordar o problema de integrar mensagens CPDLC no SIMATM. Relembrando a Figura 4, durante a análise inicial ao problema foi necessário encontrar alternativas que servissem de interface entre o sistema de terra e o simulador. Esta interface é bastante importante pois é a responsável por fazer a ligação *Data link* entre os dois sistemas sendo "a porta de entrada e saída" de todas as mensagens CPDLC no sistema de controlo de tráfego aéreo. A Figura 6 representa as duas alternativas de arquitectura estudadas, estando estas alternativas condicionadas às arquitecturas dos sistemas existentes.

A primeira alternativa que surgiu foi a de utilizar o *Data Link Processor* como porta de entrada dos dados *Data Link* no sistema terra, ignorando as componentes *Data Link Server* e ATM/*Data Link* Interface (ADI). Esta escolha teria como vantagem o facto de tanto o *Data Link Processor* como as componentes do simulador terem como protocolo de comunicação xmlRAC como mostrado na Figura 4, não havendo assim

necessidade de conversão de mensagens em diferentes protocolos, tornando a comunicação mais simples e homogénea.

A grande desvantagem desta opção é existirem componentes no sistema terra que seriam ignoradas na interacção com o simulador, logo o objectivo de tornar a simulação o mais real possível não seria conseguido.

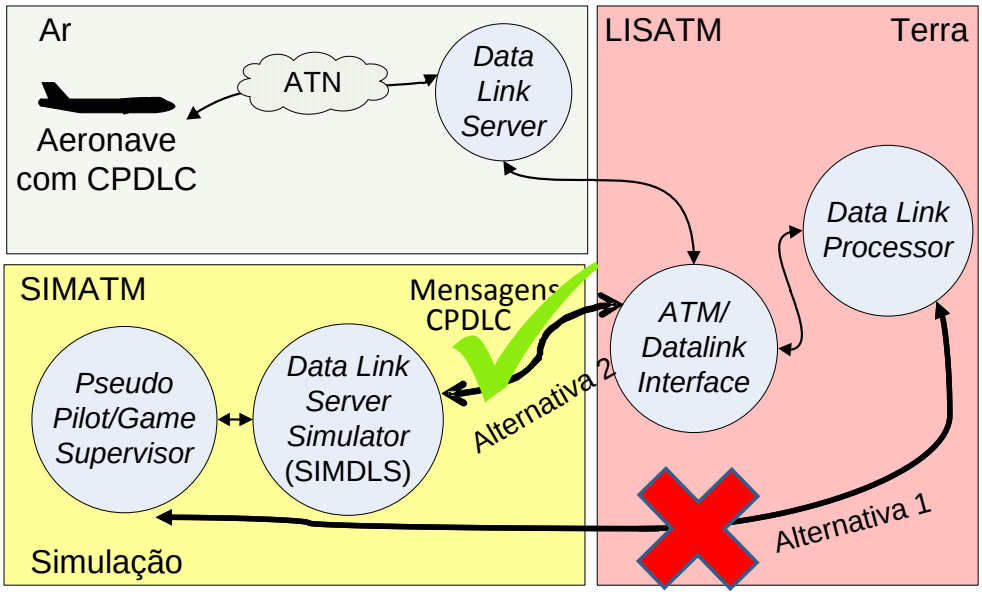

Figura 6: Alternativas de arquitectura

A segunda alternativa foi a de utilizar a componente ATM/*Data Link* Interface como porta de entrada dos dados *Data Link* no sistema terra. Esta opção teria como vantagem o facto de tornar a simulação mais real e de englobar todas as componentes internas do sistema terra na simulação e testes. O *Data Link Server* continuaria excluído, pois é uma caixa negra para o sistema terra, mas seria substituído por uma componente de simulação.

A desvantagem desta abordagem é ser necessário efectuar a conversão entre formatos de mensagens, pois as componentes do SIMATM comunicam utilizando mensagens no formato XML e a componente que vai ser a porta de entrada de dados no LISATM espera recebê-los no formato ASN.1.

3.3 Tomada de decisão

Tendo em conta os pontos apresentados na secção anterior, a decisão de como fazer a ligação entre os sistemas LISATM e SIMATM recaiu na segunda opção, ou seja, seria

desenvolvido um simulador de servidor de *Data Link* (SIMDLS) que seria servidor xmlRAC para a componente *Pseudo Pilot* e servidor TCP (*Transmission Control Protocol*) trocando mensagens no formato ASN.1 com a componente ATM/*Data Link* Interface (alternativa 2 na Figura 6). Deste modo, tal como explicado na Secção 3.2 a simulação dos comandos CPDLC através do *Data Link* será bastante mais real do que se tivesse seguido pela primeira alternativa apresentada.

A decisão por esta opção é particularmente relevante para a função de teste do sistema, uma vez que para a função de treino de controladores de tráfego aéreo é invisível para o utilizador a forma como o simulador e o LISATM são interligados.

É devido a esta decisão que surgiram as alterações na lista de tarefas apresentada na Secção 1.4 tendo passado a existir a necessidade de desenvolver de raiz uma componente para o simulador, e tendo de ser efectuado todo um ciclo de desenvolvimento de *software*. Dado que a realização deste ciclo é um processo complexo e que não estava previsto inicialmente, foi necessário retirar duas tarefas do planeamento inicial de forma a acrescentar a tarefa de desenvolvimento desta nova componente. A escolha das tarefas a retirar recaiu sobre aquelas pois seriam idênticas à tarefa de implementar mensagens CPDLC no SIMATM mas realizadas noutros sistemas, logo e tendo em conta a importância que o desenvolvimento do SIMDLS tem para o sucesso do projecto optámos por esta opção de planeamento.

3.4 Sumário

Este capítulo apresentou os conceitos relevantes e relacionados com o CPDLC sendo que de seguida expus as alternativas estudadas para efectuar a interface entre o simulador e o sistema de controlo de tráfego aéreo, as vantagens e desvantagens das mesmas, e justifiquei a decisão tomada.

No próximo capítulo irei detalhar o trabalho realizado para responder ao problema identificado segundo as linhas da decisão de arquitectura de forma a concretizar a implementação de protótipos que adicionem a capacidade CPDLC ao SIMATM.

Capítulo 4

Trabalho realizado

Ao longo deste capítulo descrevo o trabalho realizado ao longo das várias fases definidas no modelo de desenvolvimento de *software* usado na NAV Portugal E.P.E., bem como apresento os resultados obtidos em cada uma dessas fases. O trabalho realizado divide-se em dois subconjuntos: o desenvolvimento do simulador de servidor de *Data Link* e a evolução de componentes do SIMATM para suportar mensagens CPDLC.

4.1 Enquadramento no ambiente de desenvolvimento

Antes de passar ao trabalho realizado propriamente dito, nesta secção procuro enquadrar o leitor no ambiente de desenvolvimento existente na NAV Portugal, E.P.E. começando por uma descrição das ferramentas informáticas que usei, seguindo-se a apresentação do modelo de desenvolvimento de *software* que segui, da *framework* ATC que tive de interiorizar, e terminando com uma exposição da plataforma de testes usada na empresa.

4.1.1 Ferramentas de desenvolvimento de *software*

Durante a realização do projecto utilizei o Eclipse[1] para fazer a codificação de componentes na linguagem Java, no qual estava instalado o *plugin Ant* que serve para automatizar o processo de compilação de *software* e de gestão de dependências entre recursos.

De forma a conseguir controlar as várias versões do código produzido e a comparar as diferenças entre elas utilizei um repositório CVS, que também tem como objectivo permitir o acesso aos documentos que vão sendo produzidos nos projectos. O acesso aos ficheiros pode ser feito através da ferramenta TortoiseCVS[2] ou de uma plataforma interna *online*.

[1] O ambiente integrado de desenvolvimento Eclipse está disponível em www.eclipse.org.

[2] A ferramenta TortoiseCVS pode ser obtida em www.tortoisecvs.org.

Por último, utilizei uma plataforma interna denominada Trac que tem como propósito facilitar a divulgação de documentação sobre cada projecto.

4.1.2 Modelo de desenvolvimento de *software* em V

O modelo de desenvolvimento de *software* utilizado na DSTI é o modelo em V [2] que pode ser representado como mostrado na Figura 7:

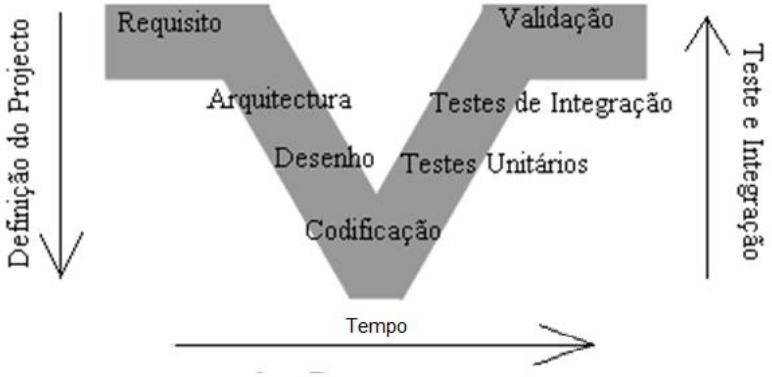

Figura 7: Modelo de desenvolvimento de *software* em V

As fases do ciclo de vida do modelo de desenvolvimento em V podem ser sucintamente explicadas da seguinte forma:

- O início do desenvolvimento é feito através do levantamento de requisitos funcionais e não funcionais do produto que se pretende desenvolver. Um requisito funcional define uma função ou serviço que um sistema ou uma sua componente deve oferecer. Um requisito não-funcional especifica critérios que podem ser usados para avaliar o modo de operação de um sistema;
- Depois levantados os requisitos passa-se à especificação da arquitectura geral do *software*. No caso específico do meu projecto, considerando que se trata da evolução de um sistema existente, alguma da arquitectura já se encontrava definida;
- De seguida, baseado nas especificações anteriores, define-se detalhadamente cada componente do *software* e seus interfaces. Esta fase pretende transformar os resultados das suas antecessoras em artefactos capazes de serem directamente interpretados na fase de codificação.

Esta primeira parte do modelo em V denomina-se definição do projecto, sendo a parte seguinte designada de teste, integração e validação.

- Terminada a codificação, iniciam-se os testes unitários, os quais têm o propósito de verificar a definição efectuada de cada componente de *software*;
- Depois de cumpridos os testes unitários passa-se aos testes de integração, com o objectivo de verificar os interfaces dos componentes e as especificações da arquitectura geral do *software*, garantindo que os componentes anteriores estão correctamente integrados;
- Por último, depois de realizados e de obter sucesso nos testes unitários e de integração, procede-se à realização dos testes de validação/aceitação, feitos com a contribuição do cliente e realizados antes da entrega do produto final.

Este modelo de desenvolvimento de *software* tem como objectivos minimizar os riscos, aumentar as garantias e qualidade do produto, reduzir os custos e duração do projecto, e aumentar a comunicação entre todos os intervenientes envolvidos. Estes objectivos são conseguidos devido à contratação clara entre fases e à transparência da estrutura e responsabilidade que este modelo de desenvolvimento permite.

Este modelo é utilizado na DSTI devido à complexidade dos projectos existentes. Assim, considerando os objectivos referidos, a utilização deste modelo é vantajosa pois os artefactos produzidos nas fases de levantamento de requisitos, arquitectura, e desenho informam a fase de codificação. Em relação aos testes, a definição feita na primeira parte do ciclo de vida do projecto, facilita a identificação do que testar e como testar. Visto isto, uma boa definição de projecto permite obter bons testes.

4.1.3 *Framework* ATC

A *framework* ATC foi desenvolvida internamente na NAV Portugal E.P.E. para que todas as componentes construídas com recurso a esta *framework* possam respeitar o mesmo padrão de arquitectura e para oferecer um conjunto de serviços tais como aquisição e apresentação de dados radar, comunicação usando o protocolo xmlRAC, cálculos de distâncias entre aeronaves, injectores de mensagens em redes de dados, entre outros.

Esta *framework* faz uso da tecnologia Java e implementa, sendo um dos mais relevantes, o padrão de arquitectura de *software Model-View-Controller* [22]. Este padrão pretende separar o domínio lógico da aplicação (*model*) da sua interface (*view*)

existindo uma ligação entre ambos, denominada *controller*. Pode ver-se na Figura 8 o fluxo existente entre as várias componentes lógicas de um *software* baseado neste padrão, sendo descrito de seguida o papel de cada componente da arquitectura:

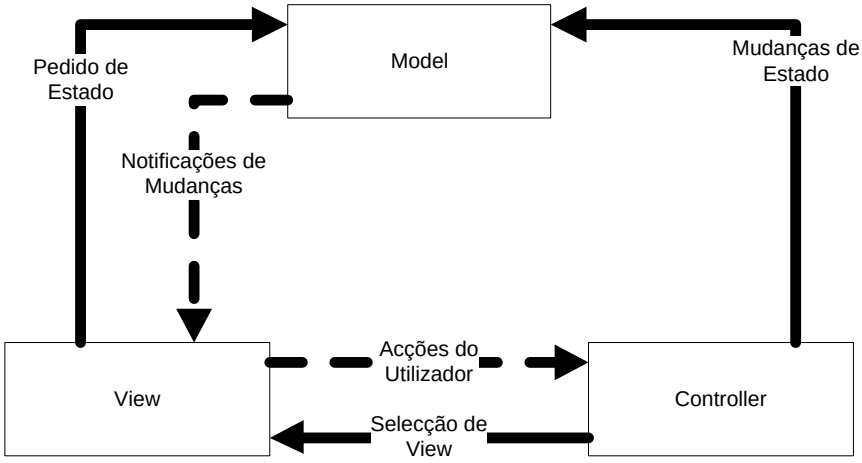

Figura 8: Padrão *Model-View-Controller*

- O modelo (*model*) é responsável por gerir o domínio lógico da aplicação que inclui a representação dos dados (usualmente guardados em base de dados) e gestão das regras de acesso aos mesmos;
- O controlador (*controller*) é responsável por traduzir as interacções do utilizador na vista em acções no modelo, por receber a resposta às acções desencadeadas no modelo, e entregar essa mesma resposta à vista correcta ajudando assim a manter um estado coerente entre os dados do modelo e os dados representados nas vistas;
- Por último, a vista (*view*) é a componente visível ao utilizador final da aplicação, correspondendo essencialmente à HMI (*Human-Machine Interface*). É da responsabilidade da vista apresentar os dados contidos no modelo.

Um exemplo de concretização do padrão *Model-View-Controller* na *framework* ATC é a apresentação da informação de vigilância. Neste caso, o modelo, denominado de *Track* representa os dados de vigilância obtidos dos radares. Estes dados são apresentados em vistas, como por exemplo, o símbolo da aeronave na posição *x,y*, a informação na *label* da aeronave que está anexada ao símbolo da aeronave, e a tabela que apresenta a informação de vigilância de todos os voos activos no momento. Por fim, cada vista

tem associado um controlador, sendo que esses controladores estão a registar as alterações no modelo *Track*.

Esta *framework* foi bastante importante na realização do projecto pois foi através da sua utilização que desenvolvi os protótipos propostos.

4.1.4 Plataforma de testes

Relembrando a Figura 7, a parte de teste e integração do modelo de desenvolvimento de *software* tem como objectivo testar o trabalho feito durante a primeira fase do modelo em V. Os testes são divididos em dois grandes grupos: unitários e de integração.

Os testes unitários são feitos sobre entidades e são concretizados recorrendo à *framework* JUnit[3].

Os testes de integração são realizados de duas formas: utilizando um injector de mensagens XML (um utilitário que acompanha a *framework* ATC) e tem como objectivo injectar mensagens pré-formatadas e em intervalos de tempo conhecidos nas componentes funcionais a testar. A utilização deste injector permite identificar erros que possam existir ainda antes de integrar as componentes umas com as outras. Depois desta integração com recurso ao injector de mensagens é feita a integração das componentes usando uma plataforma que contém duas instâncias dos sistemas SIMATM e LISATM com as devidas alterações para suportarem a funcionalidade CPDLC.

4.2 Simulador de servidor de *Data Link Air-Ground* – SIMDLS

Esta secção mostra os resultados obtidos em cada fase do modelo de desenvolvimento de *software* aplicado à concretização do simulador de servidor de *Data Link*.

4.2.1 Levantamento de requisitos

Seguidamente são apresentados os actores e uma síntese dos requisitos funcionais e não funcionais do SIMDLS (Simulador de servidor de *Data Link*).

Actores

Conteúdo confidencial.

[3] A *framework* JUnit encontra-se disponível em www.junit.org.

Requisitos funcionais

Conteúdo confidencial.

Requisitos não funcionais

Conteúdo confidencial.

4.2.2 Desenho de *software*

Conteúdo confidencial.

4.2.3 Implementação

Conteúdo confidencial.

4.2.4 Testes unitários

Conteúdo confidencial.

4.3 Evolução do SIMATM para mensagens CPDLC

O âmbito desta secção é apresentar os resultados obtidos durante as várias fases do ciclo de desenvolvimento de *software* executado para a evolução das componentes já existentes no SIMATM para suportarem o serviço CPDLC.

4.3.1 Levantamento de requisitos

A análise inicial do problema proposto levou-me a realizar um levantamento dos actores e dos requisitos funcionais e não funcionais associados ao contexto da implementação de comandos CPDLC no SIMATM.

Actores

Conteúdo confidencial.

Requisitos funcionais

Conteúdo confidencial.

Requisitos não funcionais

Conteúdo confidencial.

4.3.2 Desenho de *software*

Conteúdo confidencial.

4.3.3 Implementação

Nesta secção abordo os detalhes de implementação das componentes do SIMATM que necessitaram de ter acrescentada a funcionalidade CPDLC.

Scenario Editor

Conteúdo confidencial.

Load Scenario

Conteúdo confidencial.

Game Supervisor e *Pseudo Pilot*

Conteúdo confidencial.

4.3.4 Testes unitários

Conteúdo confidencial.

4.4 Testes de integração

Conteúdo confidencial.

4.5 Sumário

Este capítulo teve como objectivo apresentar o trabalho efectuado durante a realização do projecto em que participei. Comecei por descrever as ferramentas informáticas, o modelo de desenvolvimento de *software* existente na empresa e os objectivos de cada fase do mesmo, bem como a *framework* ATC, e a plataforma de testes utilizada.

Depois apresentei os resultados obtidos durante os ciclos de desenvolvimento das duas tarefas principais do projecto nomeadamente a concretização do simulador de servidor de *Data Link*, SIMDLS, e a implementação de mensagens CPDLC no SIMATM.

No próximo e último capítulo deste documento serão apresentadas as conclusões do projecto bem como as principais contribuições que o cumprimento do projecto teve para a NAV Portugal E.P.E. e para mim. Será também referida uma perspectiva do trabalho a realizar futuramente.

Capítulo 5

Conclusão

Neste último capítulo apresento os contributos retirados da realização do projecto, focando as principais contribuições, dificuldades encontradas, e competências adquiridas durante o estágio na NAV Portugal E.P.E..

Por fim, refiro uma perspectiva do trabalho que poderá dar seguimento ao meu projecto de estágio.

5.1 Principais contribuições

A realização deste projecto teve como principal contribuição a evolução da plataforma de simulação existente na NAV Portugal E.P.E. para que esta esteja equipada com a tecnologia CPDLC.

De forma a realizar essa evolução cumpri a maioria das fases do modelo de desenvolvimento de *software* em V. Sendo que a utilização deste modelo é complementada com boas práticas da metodologia *agile*, tive de efectuar duas iterações do mesmo, uma para o SIMDLS e outra para a evolução dos componentes já existentes no SIM-ATM.

Como resultado da análise inicial ao problema e após estudadas as alternativas de arquitectura, foi decidido implementar totalmente uma nova componente para o simulador. Esse componente tem como responsabilidade simular o comportamento do *Data Link Server*, sendo que a criação desta componente é bastante importante para que o simulador consiga exercitar o LISATM da forma mais real possível.

Uma segunda contribuição bastante importante que resulta também da realização deste projecto foi a publicação de um artigo científico [6] na 6ª Conferência Ibérica de Sistemas e Tecnologias de Informação que decorreu em Chaves durante o mês de Junho de 2011. Este artigo foi escrito por mim em conjunto com os meus orientadores de pro-

jecto e o Eng.º José Vermelhudo da NAV Portugal E.P.E. A apresentação do artigo foi realizada por mim durante os trabalhos da conferência.

5.2 Dificuldades encontradas

A maior dificuldade que senti esteve relacionada com a grande complexidade da *framework* ATC, tendo precisado de bastante estudo para conseguir começar a desenvolver código. Este estudo foi necessário pois, tendo em conta que os produtos construídos sobre esta *framework* seguem padrões específicos, é necessário que as novas funcionalidades continuem a seguir esses mesmos padrões. Também é importante que se perceba bem o funcionamento da *framework* ATC pois algumas das novas funcionalidades podem aproveitar trabalho já efectuado, não sendo necessário existir duplicação de código.

Devido à grande complexidade da *framework* esteve o facto de não ter tido tempo para concretizar a fase de testes unitários, tendo dedicado mais tempo à implementação do SIMDLS e da funcionalidade CPDLC no SIMATM.

Outra dificuldade encontrada teve a ver com o facto da realização do meu trabalho ter coincidido com a instalação do novo sistema de controlo de tráfego aéreo das torres de Faro e do Porto, o que levou a uma limitação na utilização e disponibilidade dos recursos necessários para efectuar a verificação dos requisitos.

5.3 Competências adquiridas

Durante a realização deste projecto adquiri variadas competências, sendo uma das mais relevantes o ter trabalhado num ambiente completamente diferente do que estava habituado e numa área de negócio bastante complexa. A integração na forma de trabalhar num ambiente tão exigente como o do controlo de tráfego aéreo é uma grande mais-valia que retiro deste projecto.

Outras competências adquiridas estão relacionadas com a utilização tanto do modelo de desenvolvimento em V como do padrão de arquitectura de *software Model-View-Controller*. Tendo na FCUL adquirido conhecimentos teóricos foi-me possível agora aplicar praticamente esses conhecimentos num projecto específico.

Por fim, e relacionado com a produção do artigo científico adquiri competências na escrita e apresentação do trabalho tanto a responsáveis da empresa (tendo havido uma sessão com cerca de 30 pessoas) e da comunidade científica.

5.4 Perspectiva futura

Esta secção apresenta uma perspectiva de trabalho que dê continuidade ao meu projecto. Primeiro que tudo é necessário fechar o ciclo de desenvolvimento de *software* da funcionalidade implementada. Como referido nas Secções 4.2.4 e 4.3.4 a fase de testes unitários não foi feita na totalidade faltando produzir o código dos mesmos. Também é referido na Secção 4.4 que os testes de integração não foram realizados na sua totalidade sendo que finalizar esta fase do modelo em V também é trabalho futuro.

Para a realização do projecto foram consideradas e implementadas as mensagens CPDLC mostradas na Tabela 3, sendo que o conjunto total de mensagens é bastante maior [4]. Futuramente será necessário evoluir as componentes do simulador (SIMDLS, *Pseudo Pilot*, e *Game Supervisor*) para que se consiga processar e produzir todo o conjunto de mensagens CPDLC.

Após finalizada a concretização de mensagens CPDLC para a simulação do que se passa no ar, será necessário concretizar a utilização desta tecnologia para a comunicação no solo. Este era aliás um dos objectivos deste projecto, que foi substituído no decorrer do mesmo pelo desenvolvimento do SIMDLS. Esse passo envolverá o evoluir do produto *Ground Situation Display* para que possa ser possível substituir as autorizações orais de voo por mensagens escritas, sendo que para realizar esta tarefa também será necessário evoluir o SIMDLS para que este processasse as mensagens em causa.

Referências

1. Airports Council International. World airport traffic report 2009. www.airports.org/statistics.

2. Amman, P. and Offutt, J. Introduction to software testing. *Cambridge University Press*, 2008, 3-24.

3. Bolczak, R. Operational evolution of an integrated URET/CPDLC capability. *Digital Avionics Systems Conference, vol. 2*, (2002), 7.D.3-1–11.

4. European Organisation for the Safety of Air Navigation. *EUROCONTROL specification on Data Link services*. 2009.

5. European Organisation for the Safety of Air Navigation. Air traffic statistics and forecasts. www.eurocontrol.int/statfor.

6. Ferreira, R., Dias, M., Vermelhudo, J., and Ferreira, A. Developing a controller pilot data link communication simulator. *Actas da 6ª Conferência Ibérica de Sistemas e Tecnologias de Informação*, (2011), 579-583.

7. Gaspar, P. New-generation simulator. *Air Traffic Technology International*, 2007, 72-75.

8. Gil, F.O. Dependability analysis of the controller-pilot data link communications application. *4th Latin-American Symposium on Dependable Computing*, (2009), A-14–16.

9. Kapp, V. and Classe, C. Designing an ATC CPDLC environment as a common information space. *25th Digital Avionics Systems Conference*, (2006), 6.A.6-1–12.

10. Medina-Mora, M., Sherry, L., Hoppenbrouwers, J., and Boehm-Davis, D.A. Simulated task environment for HCI analysis of NextGen data comm. *International Conference on Human-Computer Interaction in Aeronautics*, (2010).

11. Mioch, T., Mistrzyk, T., and Rister, F. Procedure design and validation by cognitive task model simulations. *19th Conference on Behavior Representation in Modeling and Simulation*, (2010), 232-239.

12. Mulkerin, T. Free flight is in the future: Large-scale controller pilot data link communications emulation testbed. *Aerospace and Electronic Systems Magazine, vol. 18, no. 9*, 2003, 23–27.

13. NAV Portugal. *SIMATM system architecture specification version 1.0*. 2006.

14. Nguyen, T.C., Bretmersky, S., and Murawski, R. Impact of CPDLC traffic loads on VHF digital link mode 3. *Digital Avionics Systems Conference, vol. 1*, (2004), 1.A.1-1–9.

15. Rakas, J. and Yang, S. Analysis of multiple open message transactions and controller-pilot miscommunications. *7th USA/Europe Air Traffic Management Research and Development Seminar*, (2007).

16. Robinson, D.C. CSMA versus prioritized CSMA for air-traffic-control improvement. *IEEE Aerospace Conference, vol. 3*, (2002), 1191–1196.

17. Shingledecker, C., Giles, S., Darby, E.R., Pino, J., and Hancock, T.R. Projecting the effect of CPDLC on NAS capacity. *24th Digital Avionics Systems Conference, vol. 1*, (2005), 2.B.5-1–8.

18. Signore, T.L. and Girard, M. The aeronautical telecommunication network (ATN). *Military Communications Conference, vol. 1*, (1998), 40–44.

19. Signore, T.L. and Hong, Y. Party-line communications in a data link environment. *19th Digital Avionics Systems Conference, vol. 1*, (2000), 2.E.4-1–8.

20. Sollenberger, R.L. and Della Rocco, P.S. Human factors issues in the collocation of URET, TMA, and CPDLC. *23rd Digital Avionics Systems Conference, vol. 1*, (2004), 5.B.5-1–9.

21. Studenberg, F. The capacity and performance of a CPDLC system using VDL mode E. *24th Digital Avionics Systems Conference, vol. 1*, (2005), 1.B.4-1–9.

22. Sun Microsystems. Java BluePrints: Model-View-Controller. http://java.sun.com/blueprints/patterns/MVC-detailed.html.

23. Yenson, S. and Rakas, J. Impacts of a mixed media air traffic control communication environment on aviation efficiency. *12th Air Transport Research Society World Conference*, (2008).

Anexo 1

Artigo apresentado na CISTI 2011

Developing a Controller Pilot Data Link Communication Simulator

Rui Ferreira, Manuel Dias, José Vermelhudo
NAV Portugal, EPE
Firstname.Lastname@nav.pt

António Ferreira
University of Lisbon
asfe@di.fc.ul.pt

***Abstract*—Radio frequencies for controller pilot communication are becoming a scarce resource due to increasing air traffic worldwide. The controller pilot data link communication (CPDLC) technology, which is already mandatory in new aircrafts, aims at reducing radio interferences and misunderstandings of commands by allowing routine voice conversations to be carried out via text messaging. This paper describes the requirements, architecture, and development of an air traffic management (ATM) system with CPDLC capabilities in the context of a national air navigation service provider. We discuss some challenges and limitations that we encountered, and highlight the importance of building the CPDLC system based upon the existing framework for ATM simulation, which is designed to support on job training and system testing.**

Air traffic management, simulator, controller pilot communication, data link, system testing, on job training.

I. Motivation

The volume of air traffic has been increasing worldwide over the past years [1] and this trend is expected to continue in the future at an annual rate of about 3%, meaning that in 2030 there will be almost twice the number of flight movements than today [4]. This situation poses upcoming safety problems to air traffic management (ATM) because radio frequencies used in today's voice-based controller pilot communication are becoming a scarce resource. One reason for this is that when the number of aircrafts flying in a sector (or geographical area) reaches a limit, the sector is split in two, therefore requiring an extra radio frequency that is assigned to a new controller.

Naturally, a consequence of the scarcity of radio frequencies is that more people have to share the same voice channel, thus increasing the risk of communication problems, which are a significant source of operational errors and pilot deviations, due to noise and interferences, ambiguous wording, message complexity, and other factors [18]. Furthermore, since controller pilot confusions are typically resolved by repeating messages, the tendency is for even worse frequency congestion.

To tackle the scarcity of radio frequencies and increase the operational capacity of the air national service providers, or ANSPs, responsible for the safe, orderly, and rapid flow of air traffic, the International Civil Aviation Organization (ICAO) and the European Commission, through the European Organisation for the Safety of Air Navigation (EUROCONTROL), have approved the controller pilot data link communication (CPDLC) technology, which offers text messaging over digital communication channels for connecting aircrafts and control centres via the Aeronautical Telecommunications Network, or ATN [22]. The CPDLC technology should gradually replace some of the routine, non-critical, voice messages exchanged between pilots and controllers during operations on the ground and in the air. The following are some of the main advantages of using CPDLC messages for air traffic management:

- Increased communication channel capacity, because fewer bits are needed to transfer text messages than the equivalent verbal messages [20]. Also, retransmissions requests by pilots or controllers are almost unnecessary, even if the radio channel is noisy, because errors in digital text messages can be automatically detected and corrected [17];
- Increased sector productivity and capacity, due to the possibility of aircrafts periodically transmitting status reports that otherwise would have to be requested and answered via voice conversations [8], and also from the simplification of routine procedures, for instance, by assigning recurrent messages to dedicated buttons [28]. This reduced need for verbal communication has been linked to lower controller workload, meaning that up to about 15% more aircrafts can be safely managed in a single sector [20].

Both of these CPDLC advantages mitigate the scarcity of radio frequencies by better utilizing existing resources, with further optimizations in data link systems indicating that air traffic in Europe should be supported up to the year 2050 [24].

To further emphasize the importance of the CPDLC technology in air traffic management, the European Commission, through the EUROCONTROL, has approved an implementing rule that mandates the installation of digital link systems in all aircrafts built since 2011, and the same applies from 2013 onwards to ANSP organizations that have adhered to the Link-2000+ programme [5].

This paper describes the requirements, architecture, and development of an ATM simulator with CPDLC capability, which is designed for on job training and system testing in the context of a national air navigation service provider. We focus here on a subset of the messages, related to the en route flight segment (high altitude, non-stop, flights), which is less risky compared to landings and take-offs and, thus, more appropriate for the initial operations in the production ATM system.

The paper is organised as follows: in the next section we present the related work, namely other uses of simulators concerning CPDLC, most of which for evaluating network performance; in Section III we describe the framework we use for ATM simulation and in Section IV we explain the integration of CPDLC capabilities in the simulator; in Section V we discuss some challenges and limitations we encountered; and in the last section, we end with the conclusions and future work.

II. Related Work

The CPDLC technology has been maturing over the past years, having captured the interest of researchers who have used computer simulators for a variety of tasks, such as:

- Workload evaluation, based upon the execution of cognitive task models in predefined scenarios of communication, which have predicted imbalances between pilot and co-pilot workload because of the change from the aural to the visual modalities [14] and increased controller efficiency in a mixed visual/aural environment [28];
- Usability evaluation, using high or low-fidelity prototypes that replay air traffic flow scenarios to uncover issues in the human-computer interaction [13], such as the screen getting cluttered when CPDLC capabilities are integrated in a decision support tool [23] or the increased risk of controllers executing redundant actions when text messages replace voice conversations [11];
- Network performance evaluation, in which CPDLC messages are inserted in the communication network to verify that adding a new type of message to the protocol continues to satisfy the overall transfer delay requirements [21], to demonstrate that a protocol is more efficient than another [19], or to show the capacity of a proposed data link configuration can support future traffic demand [24];
- Operational evaluation, accepting inputs from human pilots and controllers during the simulation. We found three such simulators in the literature: Advanced Communications for ATM, AC/ATM [15], User Requested Evaluation Tool with CPDLC, URET/CPDLC [3], and Communication, Navigation, Surveillance for ATM, or CNS/ATM [8].

The three simulators in the previous point are closest to our work as they can potentially be used for both on job training and system testing due to the participation of pilots and controllers. However, to the best of our knowledge, research has focused on network performance and usability evaluations.

As a matter of fact, the stated objective of CNS/ATM is to assess CPDLC in an air traffic network pushed to its extreme conditions [8], and a major goal of AC/ATM is to estimate the number of aircraft that can safely operate on a single frequency, with much emphasis on the simulator supporting up to 160 aircrafts simultaneously [15]. Finally, URET/CPDLC research reports on usability changes due to the integration of CPDLC capabilities in the existing URET decision support tool [3,23].

From this review of the state of the art, we take the opportunity to highlight the importance of researching ATM systems design and development considering the synergies between on job training and system testing, as we describe next.

III. Framework for ATM Simulation

As mentioned earlier, the Link2000+ programme coordinates the application of CPDLC systems in ANSPs from 2013 onwards. But, as new aircrafts now have digital data links, and given the advantages of CPDLC, it is in the best interest of ANSPs to start the development ATM systems with CPDLC capabilities rapidly. In this section we describe work conducted in an ANSP to accommodate the required changes using an in-house framework for ATM simulation, called SIMATM.

A major objective of SIMATM is to leverage the synergy between on job training and system testing [7]:

- On the one hand, trainees have access to a replica of a real ATM system, called LISATM, which is clearly desirable;
- On the other hand, the events generated by the trainees while interacting with LISATM can actually be part of the system testing, complementing static, scripted, scenarios.

In the ATM simulators stated in the related work this synergy is less effective (or non-existent) because human controllers use high/low-fidelity prototypes instead of real systems.

A. Framework Components

SIMATM is a component-based Java framework for ATM simulation and is designed for extensibility by adhering to the model-view-controller design pattern [26]. Fig. 1 shows a data flow diagram based upon the SIMATM architecture specification [16], illustrating its main internal components as well as the interaction with the LISATM external entity.

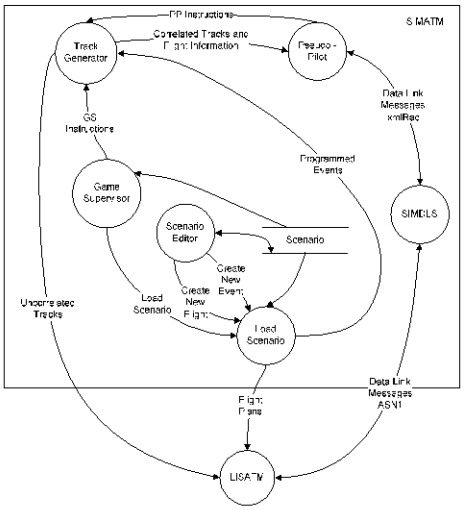

Figure 1. SIMATM components and interaction with LISATM.

Each component in SIMATM is responsible for providing a human-machine interface (HMI) and for assuring the integrity of its own data. Each of the components is described next:

- Scenario Editor: Allows the creation and updating of simulated exercise scenarios. Each scenario is stored in XML format in a database.
- Load Scenario: Converts scenarios into flight plans, which are used by LISATM (and seen by human controllers), and also by the Track Generator and Pseudo-Pilot.
- Track Generator: Calculates the next position of aircrafts considering the information in the scenario and commands issued by the Pseudo-Pilot and Game Supervisor. It also feeds LISATM with radar data.

- Pseudo-Pilot: Allows control of each aircraft in the exercise. It can also behave as an autopilot, in which case the navigation is based upon information in the scenario.
- Game Supervisor: For manual control of the exercise.
- Simulated Data Link Server (SIMDLS): Establishes a data link connection with LISATM and transmits messages between Pseudo-Pilots and the air traffic controllers.

Fig. 1 shows that some SIMATM components communicate with LISATM, providing the same information as would happen with real aircrafts and surveillance systems, thus the need for representing data and commands using the industry standard ASN.1 notation [12]. Within SIMATM, all communication between the components is supported by our own xml-Rac protocol, which is based upon XML and implemented over UDP sockets but ensuring the delivery of messages [27].

B. *Sample Simulation Scenario*

The following paragraphs describe the steps for setting up and running an ATM exercise in SIMATM:

1. The exercise scenario is created using the Scenario Editor, and this must be done offline;
2. The operator of the Game Supervisor loads the scenario. The loading itself is done by Load Scenario, which sends the scenario in flight plan format to LISATM, Pseudo-Pilot, Game Supervisor, and Track Generator, including as well flight plan *simulations*, which add navigation information and pre-programmed events (such as a change in aircraft altitude or direction) to the flight plans;
3. After the scenario is loaded, the Game Supervisor operator assigns flights defined in the scenario to each participating Pseudo-Pilot according to the respective radio frequencies. There can also be cases in which flights are automatically controlled by Track Generator instead of a Pseudo-Pilot;
4. After the flight assignment is done, the Game Supervisor can start the exercise. At this point the Track Generator produces flight information based upon the flight plans and the commands (such as flight level and speed adjustments) inserted by the Pseudo-Pilot or Game Supervisor;
5. When the Game Supervisor ends the exercise, all loaded information is cleared, and the component returns to its initial unloaded state. To repeat the exercise, the scenario file must load again (see step 2).

These steps illustrate the interactions between the components of SIMATM, and with the LISATM external entity, and provide a context for better understanding the changes needed for CPDLC integration in SIMATM.

IV. Integrating CPDLC in SIMATM

In this section we describe the integration of CPDLC capabilities into SIMATM, focusing on an illustrative subset of the messages, namely the climb/descend clearances that occur during the en route, high altitude, flight segment (see Table I).

The complete set of messages approved for the Maastricht upper area control, which pioneered the use of CPDLC in Europe, additionally includes radio frequency changes, turns and headings changes, microphone checks, and more [25].

TABLE I. Climb/Descend Clearance CPDLC Messages

Direction	Name	Description
Controller to pilot	MAINTAIN [level]	Maintain [level] altitude
	CLIMB TO [level]	Climb to [level] altitude
	DESCEND TO [level]	Descend to [level] altitude
Pilot to controller	REQUEST [level]	(Similar to the messages above, but initiated by the pilot, who must wait for the controller response)
	REQUEST CLIMB TO [level]	
	REQUEST DESCEND TO [level]	
	WILCO	Understood and will comply
Both directions	UNABLE	Cannot comply
	STANDBY	Further message will follow

From the subset of CPDLC messages in Table I we enumerate functional and non-functional requirements that lead to modifications in some of the SIMATM components. However, before that we introduce the software development process that we follow in-house.

A. *Software Development Process*

The project to integrate CPDLC capabilities in SIMATM, much like our other assignments to build complex and critical systems, follows a documented software development process, which is based upon the V-model [2] complemented with good practices from agile methodologies [9,10].

In a nutshell, besides the expected division of the process into phases (namely the requirements gathering, architecture, detailed design, coding, unit testing, integration testing, and validation), in which the outputs of each phase feed the inputs of the next ones, we also identify the functionalities that are more likely to suffer modifications.

Then, for each of these individual functionalities we apply a complete development cycle to ensure that what is built is what the client specified and so that the client can periodically validate our work. This means the system is developed incrementally to guarantee that the adaptation to changes has a minimal impact on the project planning and satisfies the client.

Next, we show outcomes from the requirements gathering phase, specific to the messages in Table I, and considering the context provided by the SIMATM components in Fig. 1.

B. *Functional Requirements*

The integration of CPDLC capabilities in SIMATM determines three functional requirements that should be satisfied during both the setting up and execution of an ATM exercise:

- The Scenario Editor should be able to create an exercise scenario containing aircrafts with CPDLC capabilities;
- The Pseudo-Pilot should be able to see if an aircraft under his/her own control has CPDLC capabilities;
- The Game Supervisor should be able to see, at any instant of the exercise, if a specific aircraft has CPDLC capabilities, and check if they are currently active.

Next, we present the behaviour that should be satisfied only while an exercise is running (not paused nor being setup):

- When SIMDLS receives a climb/descend clearance from LISATM, it should forward the information contained in the message to the Pseudo-Pilot, which presents it in the HMI. SIMDLS should allow the Pseudo-Pilot to respond using WILCO, STANDBY, and UNABLE messages;
- SIMDLS should allow Pseudo-Pilot to send REQUEST type messages to LISATM and forward the controller's reply, UNABLE or STANDBY. It should also allow the Pseudo-Pilot to send a final reply, WILCO or UNABLE.

Before the messages in Table I can be processed by SIMATM, and especially by LISATM, in which controllers use real equipment, there are session initiation and termination messages (similar to logons and logoffs) that must be processed to determine if the CPDLC capabilities of an aircraft are currently active. The details of these were omitted for simplicity sake.

C. *Non-Functional Requirements*

Besides the behaviour allowed by SIMATM it is important to understand the qualities inherent to the operation of the system. To this end, we first introduce two types of timeouts that can occur in a CPDLC context:

- Operational Timeout: When the Pseudo-Pilot does not respond with a WILCO or UNABLE message within 120 seconds after the controller in LISATM has sent a message to the aircraft, or 120 seconds after the controller has received a STANDBY message from the Pseudo-Pilot;
- Controller Timeout: When the controller in LISATM does not respond with at least an UNABLE message within 120 seconds after a REQUEST message initiated by the Pseudo-Pilot, or after 120 seconds upon sending a STANDBY message to the Pseudo-Pilot.

The clocks watching both types of timeouts are reset when a STANDBY message is received. From the two definitions of timeout, it is now possible to fully describe the non-functional requirements of a CPDLC-enabled SIMATM:

- Performance: Messages should take less than one quarter of a timeout duration (operational or controller) to be delivered from LISATM to the Pseudo-Pilot and vice-versa;
- Reliability: When a message is incorrectly received or a timeout happens, SIMATM must consider the data communication link is corrupt and should shut it down;
- No Duplication: Messages sent by LISATM should be received only once by SIMATM, irrespective of the noise in the radio communication channel;
- No Creation: Messages should be uniquely identified, and also contain the identifier of either the aircraft or the controller that originally created the message; in other words, extra messages should not be created;
- Integrity: Messages received from or sent to LISATM should not be modified by any SIMATM component;
- Usability: The interaction with the HMI of the Pseudo-Pilot must be accomplished in at most four steps, and the previous steps should always be indicated so that the human pilot can rapidly restore the context; in addition, after the insertion of a command by a pilot, SIMATM must ensure the performance requirement is accomplished;
- Capacity: SIMATM should provide all requested CPDLC capabilities to aircrafts participating in an exercise.

All requirements for CPDLC were defined in terms of the SIMATM framework and its interaction with LISATM, which naturally lead to changes in the existing components.

D. *Changes to the SIMATM Components*

In order to satisfy the identified requirements it was necessary to add CPDLC functionality to several of the SIMATM components (see Fig. 1 for reference).

Starting with the Scenario Editor, it has to support the creation of scenarios with aircrafts having CPDLC capabilities. This involved modifying the representation of the details of a scenario in the database and the corresponding changes to the HMI, such as the creation of new forms to be filled out.

Regarding SIMDLS, it handles all data link transfers between LISATM and SIMATM, and so an upgrade to CPDLC messaging was needed, namely to convert the new messages in ASN.1 format to xmlRac and vice-versa. In addition, new types of log entries were defined so that all message exchanges could continue to be recorded. This log is used to facilitate the detection of software errors and also to make it possible for the Game Supervisor to gather statistics about the types of messages most frequently used during an exercise.

Another responsibility of the Game Supervisor is to monitor the exercise, and in this matter s/he can inspect the CPDLC capabilities available in each aircraft using the HMI.

Finally, the Pseudo-Pilot component needed several new functionalities, especially in the HMI, which features additional menus and forms for creating and responding to CPDLC messages, and also to initiate and terminate data link sessions (briefly mentioned in Section IV.B).

V. Discussion and Limitations

We now discuss some challenges and limitations encountered during the integration of CPDLC capabilities in SIMATM. We begin with an explanation about the advantages and disadvantages to system testing of having the SIMDLS component assume LISATM is an external entity, when, in fact, it is the same ANSP that develops and maintains both systems. Then we describe the implications of the ICAO 24-bit aircraft identifier [6] having so far being overlooked, and how that affects on job training.

A. *Communication between SIMATM and LISATM*

As explained earlier, the SIMDLS component is responsible for forwarding messages between the SIMATM infrastructure and the LISATM external entity. In doing so it needs to convert messages from ASN.1 to xmlRac formats, and vice-versa (see Fig. 1). Alternatively, and since we have access to the LISATM internals, which actually uses the same xmlRac protocol for its inter-component communication, we could have avoided the data conversion altogether by directly linking SIMDLS to a so-called data link processor (or DLP) in LISATM, which is itself very close to the HMI operated by the human controller. This approach would simplify the design of SIMDLS and reduce the complexity of implementing CPDLC messages to only one format.

However, the main disadvantage of directly linking SIMDLS to an internal component of LISATM is that integration testing and validation would not stress the full data link connection that exists between real aircrafts and LISATM, which relies on messages in ASN.1 format. In addition, the SIMDLS data link simulator represents an all-purpose communication channel defined in the ATN [22], so it goes beyond CPDLC messaging. Again, it is advantageous to test the entire chain of components for the other types of messages and services proposed by the air traffic management industry.

B. *Lack of Unique Aircraft Identifier*

An important element of ATM simulation scenarios is the flight plan, which, among other blocks of data, describes the time of departure and arrival from/to airports, and, particularly, the aircraft identification, usually in the form of a callsign. The traditional format of a callsign is two or three letters followed by the flight number (which controllers always use to refer to an aircraft in voice-based communication), but recently ICAO has recommended the adoption of a 24-bit identifier to guarantee that only one aircraft receives its designated CPDLC messages, even if it happens that the same callsign is wrongly used by two airborne aircrafts [6].

However, as it currently stands, SIMATM does not feature the ICAO 24-bit aircraft identifier, even more so because the related LISATM flight data processing system (FDPS) is still being upgraded, a process that started after the 24-bit requirement became stable.

VI. Conclusions and Future Work

The integration of CPDLC capabilities in an ATM simulator facilitates on job training for novice and expert controllers in the upcoming transition from voice to text-based communication with aircraft pilots, essential to ease the scarcity of radio frequencies. To this end, we described the modifications made to the components of an in-house simulation framework, SIMATM, and highlighted the importance of running and testing it together with a real air traffic management system.

Our current priority is to incorporate the ICAO 24-bit aircraft identifier in the flight plans and in the HMIs of the relevant components, as this improves the realism of the exercise scenarios; in fact this will be a feature of the next version of SIMATM. We also plan to go beyond the en route flight segment and into the more challenging airport approach and ground operations, but that will happen further down the road.

References

[1] Airports Council International, "World Airport Traffic Report 2009," www.airports.org/statistics, accessed Feb. 2011.

[2] P. Ammann and J. Offutt, Introduction to Software Testing. New York, NY, USA: Cambridge University Press, 2008, pp. 3–24.

[3] R. Bolczak, "Operational evolution of an integrated URET/CPDLC capability," in Proceedings of the 21st Digital Avionics Systems Conference, vol. 2. Irvine, CA, USA, 2002, pp. 7.D.3-1–11.

[4] European Organisation for the Safety of Air Navigation, "Air traffic statistics and forecasts," www.eurocontrol.int/statfor, accessed Feb. 2011.

[5] European Organisation for the Safety of Air Navigation, "Link 2000+ Programme," www.eurocontrol.int/link2000, accessed Feb. 2011.

[6] European Organisation for the Safety of Air Navigation, "Mode S technical overview," www.eurocontrol.int/msa/public/standard_page/modes_tech_overview.html, accessed Feb. 2011.

[7] P. Gaspar, "New-generation simulator," in Air Traffic Technology International, 2007, pp. 72–75.

[8] F.O. Gil, "Dependability analysis of the controller-pilot data link communications application," in Proceedings of the 4th Latin-American Symposium on Dependable Computing. João Pessoa, Paraíba, Brazil, 2009, pp. A-14–16.

[9] R. Hightower and N. Lesiecki, Java Tools for eXtreme Programming. New York, NY, USA: Wiley, 2002.

[10] C. Murphy, "Improving application quality using test-driven development," www.methodsandtools.com/archive/archive.php?id=20, accessed Feb. 2011.

[11] V. Kapp and C. Classe, "Designing an ATC CPDLC environment as a common information space," in Proceedings of the 25th Digital Avionics Systems Conference. Portland, OR, USA, 2006, pp. 6.A.6-1–12.

[12] J. Larmouth, ASN.1 Complete. San Francisco, CA, USA: Morgan Kaufmann, 1999.

[13] M. Medina-Mora, L. Sherry, J. Hoppenbrouwers, and D.A. Boehm-Davis, "Simulated task environment for HCI analysis of NextGen data comm," in Proceedings of the International Conference on Human-Computer Interaction in Aeronautics. Cape Canaveral, FL, USA, 2010.

[14] T. Mioch, T. Mistrzyk, and F. Rister, "Procedure design and validation by cognitive task model simulations," in Proceedings of the 19th Conference on Behavior Representation in Modeling and Simulation. Charleston, SC, USA, 2010, pp. 232–239.

[15] T. Mulkerin, "Free flight is in the future: Large-scale controller pilot data link communications emulation testbed," in Aerospace and Electronic Systems Magazine, vol. 18, no. 9, 2003, pp. 23–27.

[16] NAV Portugal, "SIMATM system architecture specification," version 1, Aug. 2006.

[17] T.C. Nguyen, S. Bretmersky, and R. Murawski, "Impact of CPDLC traffic loads on VHF digital link mode 3," in Proceedings of the 23rd Digital Avionics Systems Conference, vol. 1. Salt Lake City, UT, USA, 2004, pp. 1.A.1-1–9.

[18] J. Rakas and S. Yang, "Analysis of multiple open message transactions and controller-pilot miscommunications," in Proceedings of the seventh USA/Europe Air Traffic Management Research and Development Seminar. Barcelona, Spain, 2007.

[19] D.C. Robinson, "CSMA versus prioritized CSMA for air-traffic-control improvement," in Proceedings of the IEEE Aerospace Conference, vol. 3. Big Sky, MT, USA, 2002, pp. 1191–1196.

[20] C. Shingledecker, S. Giles, E.R. Darby, J. Pino, and T.R. Hancock, "Projecting the effect of CPDLC on NAS capacity," in Proceedings of the 24th Digital Avionics Systems Conference, vol. 1. Washington, DC, USA, 2005, pp. 2.B.5-1–8.

[21] T.L. Signore and Y. Hong, "Party-line communications in a data link environment," in Proceedings of the 19th Digital Avionics Systems Conference, vol. 1. Philadelphia, PA, USA, 2000, pp. 2.E.4-1–8.

[22] T.L. Signore and M. Girard, "The aeronautical telecommunication network (ATN)," in Proceedings of the IEEE Military Communications Conference, vol. 1. Boston, MA, USA, 1998, pp. 40–44.

[23] R.L. Sollenberger and P.S.Della Rocco, "Human factors issues in the collocation of URET, TMA, and CPDLC," in Proceedings of the 23rd Digital Avionics Systems Conference, vol. 1. Salt Lake City, UT, USA, 2004, pp. 5.B.5-1–9.

[24] F. Studenberg, "The capacity and performance of a CPDLC system using VDL mode E," in Proceedings of the 24th Digital Avionics Systems Conference, vol. 1. Washington, DC, USA, 2005, pp. 1.B.4-1–9.

[25] V. Stuhlsatz, "Air/ground data link procedures for flights within the area of responsibility of Maastricht-UAC." European Organization for the Safety of Air Navigation, 2008.

[26] Sun Microsystems, "Java blueprints: guidelines, patterns, and code for end-to-end Java applications," www.oracle.com/technetwork/java/javaee/blueprints/index.html, accessed Feb. 2011.

[27] NAV Portugal, "xmlRac interface design document," version 1, Apr. 2008.

[28] S. Yenson and J. Rakas, "Impacts of a mixed media air traffic control communication environment on aviation efficiency," in Proceedings of the 12th Air Transport Research Society World Conference. Athens, Greece, 2008.